UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/561,941	02/15/2006	Takeshi Inuo	029471-0194	3034

22428 7590 09/24/2008
FOLEY AND LARDNER LLP
SUITE 500
3000 K STREET NW
WASHINGTON, DC 20007

EXAMINER	
VICARY, KEITH E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
09/24/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/561,941	INUO, TAKESHI	
	Examiner	Art Unit	
	Keith Vicary	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 July 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,5-18 and 20-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 5-18 and 20-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1, 5-18 and 20-22 are pending in this office action and presented for examination. Claims 1, 5, 12, 14, 16, 18, and 20-22 are newly amended and claims 3-4 are newly cancelled by amendment filed 7/7/2008.

Claim Objections

2. Claim 16 is objected to because of the following informalities. Appropriate correction is required.

a. Claim 16 appears to have a space before a comma in line 15.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1, 5-18 and 20-22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

5. Claims 1, 12, 14, 16, 18, 20, and 21 recite the limitation “said command sequence implementation procedure setting a time period for loading program data of the processing device or making the program data valid." The instant specification does

not appear to disclose a command sequence implementation procedure setting a time period for loading program data of the processing device or making the program data valid. The examiner has looked closely over figures such as Figure 25 and the accompanying relevant portion of the specification, which disclose of certain actions being executed at points of time, but there does not appear to be any disclosure in which a command sequence implementation procedure actually sets a time period for loading program data of the processing device or making the program data valid. Examiner advises applicant to explicitly point out where in the specification this limitation is supported.

b. Claims 5-11, 13, 15, 17, and 22 are rejected for failing to alleviate the rejection of their base claims above.

6. Claim 1 recites the limitation “a plurality of program data memories, each holding a program that creates a logic circuit directly in said reconfigurable hardware for each of said processing units…said program is given control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs…said program executes a command sequence implementation procedure for translating said command sequence intermediate code into a data string that can be executed by the control device.” A single program which meets all of the above limitations does not appear to be present in the original disclosure.

7. Claim 1 recites “said command sequence implementation procedure executing a program data generation procedure in which the operation content of a processing unit

is translated into a data string that can be executed by the processing device" in the last three lines. A command sequence implementation procedure which itself executes a program data generation procedure does not appear to be present in the original disclosure.

c. Claims 5-11 are rejected for failing to alleviate the rejection of claim 1 above.

8. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

9. Claims 1, 5-11, 14-18 and 20-22 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

10. Claim 1 recites the limitation "includes a command" in line 22; it is indefinite as to what exactly includes a command.

11. Claim 1 recites the limitation "said program is given control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs" in lines 24-26; it is indefinite as to why a program which creates a logic circuit directly in said reconfigurable hardware would be given control flow of the application program, completion data, structural information, and a plurality of command sets as inputs, as it

appears that the program is *generated* with the aforementioned inputs and do not take in the aforementioned inputs itself.

12. Claim 1 recites the limitation “executes control flow analysis procedure for dividing the application program into a plurality of said processing units and generating a command sequence intermediate code” in lines 26-28. It is indefinite as to what executes control flow analysis procedure. If it is the aforementioned program, it is indefinite as to how a program which creates a logic circuit in reconfigurable hardware can itself execute control flow analysis procedure for dividing the application program into a plurality of said processing units and generating a command sequence intermediate code. In other words, it is indefinite as to how a division of a program somehow divides a bigger application program into divisions of a program.

13. Claim 1 recites the limitation "the application program" in line 24. There is insufficient antecedent basis for this limitation in the claim.

14. Claim 1 recites the limitation “said program executes a command sequence implementation procedure for translating said command sequence intermediate code into a data string that can be executed by the control device" in lines 29-31. It is indefinite as to why a program which creates a logic circuit directly in said reconfigurable hardware would execute a command sequence implementation procedure for translating said command sequence intermediate code into a data string that can be executed by the control device.

15. Claim 1 recites “said command sequence implementation procedure executing a program data generation procedure in which the operation content of a processing unit

is translated into a data string that can be executed by the processing device” in the last three lines. It is indefinite as to how a command sequence implementation procedure itself executes a program data generation procedure.

d. Claims 3-11 are rejected to alleviate the rejection of claim 1 above.

16. Claim 14 recites the limitation “said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units and a control device" in lines 3-5. It is indefinite as to whether the control device is merely part of the electronic computer along with the processing device, or whether the processing device with reconfigurable hardware can create a logic circuit for said control device.

17. Claim 14 recites the limitation “and having a program” in line 9. It is indefinite as to what is "having a program," as this does not appear to be a proper method step. If the limitation is describing an aspect of some hardware, the hardware should be explicitly recited.

18. Claim 14 recites the limitation “executing a control flow analysis procedure for generating a command sequence executed by said control device” in lines 12-13. It is indefinite as to whether the command sequence is executed by said control device, or whether the control flow analysis procedure is executed by said control device.

e. Claim 15 is rejected for failing to alleviate the rejection of claim 14 above.

19. Claim 16 recites the limitation “said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units and a control device" in lines 3-5. It is indefinite as to whether the control device is merely part of the electronic computer along with the processing device, or whether the processing device with reconfigurable hardware can create a logic circuit for said control device.

20. Claim 16 recites the limitation “executing a control flow analysis procedure for generating a command sequence executed by said control device” in lines 12-13. It is indefinite as to whether the command sequence is executed by said control device, or whether the control flow analysis procedure is executed by said control device.

f. Claim 17 is rejected for failing to alleviate the rejection of claim 16 above.

21. Claim 18 recites the limitation “said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units and a control device" in lines 3-4. It is indefinite as to whether the control device is merely part of the electronic computer along with the processing device, or whether the processing device with reconfigurable hardware can create a logic circuit for said control device.

22. Claim 18 recites the limitation "the command sequence intermediate code" in line 13. There is insufficient antecedent basis for this limitation in the claim.

23. Claim 18 recites the limitation "the control flow of the application program" in the third to last line. There is insufficient antecedent basis for this limitation in the claim, as although a control flow in general had been previously recited, a control flow of the application program in particular had not been previously recited.

24. Claim 20 recites the limitation “said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units and a control device" in lines 3-5. It is indefinite as to whether the control device is merely part of the electronic computer along with the processing device, or whether the processing device with reconfigurable hardware can create a logic circuit for said control device.

25. Claim 20 recites the limitation “and having a program” in line 9. It is indefinite as to what is "having a program," as this does not appear to be a proper method step. If the limitation is describing an aspect of some hardware, the hardware should be explicitly recited.

26. Claim 20 recites the limitation “executing a control flow analysis procedure for generating a command sequence executed by said control device” in lines 12-13. It is indefinite as to whether the command sequence is executed by said control device, or whether the control flow analysis procedure is executed by said control device.

27. Claim 21 recites the limitation “said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units and a control device" in lines 3-5. It is indefinite as to whether the control device is merely part of the electronic computer along with the processing device, or whether the processing device with reconfigurable hardware can create a logic circuit for said control device.

28. Claim 21 recites the limitation “executing a program data generation procedure for generating program data executed by the processing device” in lines 19-20. It is indefinite as to whether it is the program data generation procedure or the program data which is executed by the processing device.

g. Claim 22 is rejected for failing to alleviate the rejection of claim 21 above.

29. Claim 22 recites the limitation “the completion” in line 7. There is insufficient antecedent basis for this limitation in the claim. Note that this rejection was present in paragraph 14 of the previous office action and is thus not a new grounds of rejection.

Claim Rejections - 35 USC § 102

30. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

31. Claim 18 is rejected under 35 U.S.C. 102(b) as being anticipated by Smith et al. (Smith) (US PAT 6658564).

32. **Consider claim 18**, Smith discloses a program generation method for an electronic computer executing an application program divided into a plurality of processing units (col. 2, lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data), wherein said electronic computer includes a processing device with reconfigurable hardware that can create a logic circuit for each of said processing units (col. 2, lines 1-8, configuration data, programmable logic device) and a control device (col. 5, lines 49-57, either arrangement that implements the VC-OS, which may be a microprocessor or a programmable logic device), comprising: analyzing a control flow (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths; col. 11, lines 1-3, analyze critical paths as above in order to assign partitions); implementing a command sequence procedure in which a command sequence is generated by translating the command sequence intermediate code into a form that can be executed by the electronic computer (col. 11, lines 57-63, compiling software functions into threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler; col. 12, line 1, final executable code image; also see col. 12, lines 1-8, main function and dynamically linked functions), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable

hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions); generating program data in which operational content of a processing unit is translated into a form that can be executed by the electronic computer (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler) wherein the application program is divided so that each processing unit can be stored in a program data memory that holds a program creating a logic circuit for each processing unit in said reconfigurable hardware (col. 2, lines 1-8, configuration data) when the control flow of the application program is analyzed and divided into processing units in said control flow analysis procedure (col. 2, lines 18-20, software development tools; col. 8, lines 50-53 and 58-61, allocates programmable logic resources to functions; col. 11, lines 1-3; system design language profiler; also see above).

Claim Rejections - 35 USC § 103

33. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

34. Claim 1, 5, 15-17 and 21-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside et al. (Fallside) (US PAT 6326806) in view of Smith in view of Hanrahan et al. (Hanrahan) (US 6288566).

35. **Consider claim 1**, Fallside discloses a processing device (Figure 1 as a whole, specifically including the FPGA 104 and the configuration control circuit 106) and a control device (Figure 1, configuration control circuit 106), said processing device including reconfigurable hardware for processing units (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA), wherein said processing device comprises: a processing element with reconfigurable hardware (Figure 1, FPGA 104), a program that creates a logic circuit directly in said reconfigurable hardware (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA), said control device executing a command specified by the processing device (col. 4, lines 27-29, initiate reconfiguration), wherein said command is instructed to be executed when the processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions) and includes a command for execution of switching programs logically creating the reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of dividing an application program into a plurality of processing units and generating program data and command code sequences, and a program, wherein said program is given control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs, executes a control flow analysis

procedure for dividing the application program into a plurality of said processing units and generating a command sequence intermediate code, said program executes a command sequence implementation procedure for translating said command sequence intermediate code into a data string that can be executed by the control device, said command sequence implementation procedure setting a time period for loading program data of the processing device of making the program data valid; and said command sequence implementation procedure executing a program data generation procedure in which the operational content of a processing unit is translated into a data string that can be executed by the processing device. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with. Fallside also does not disclose of a plurality of program data memories, each holding a program that creates a logic circuit directly in said reconfigurable hardware for each of said processing units, and an effective block selection unit that connects one of said program data memories to said processing element.

On the other hand, Smith does disclose of dividing an application program into a plurality of processing units (col. 2, lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data) and generating program data and command code sequences (col. 1, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler; col. 12, lines 1-8 discloses of both a main function which determines the order of the dynamically linked functions),

and a program, wherein said program is given control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of the electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are essentially input into the compiler and linker), said program executes a command sequence implementation procedure for translating said command sequence intermediate code into a data string that can be executed by the control device (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler; col. 1, lines 1-8, main

function), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), said command sequence implementation procedure executing a program data generation procedure in which the operational content of a processing unit is translated into a data string that can be executed by the processing device (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into portions of configuration data, then swapping configuration data in and out of reconfigurable hardware in order to execute the whole program , and Fallside discloses of an overall hardware environment in which a control device swaps configuration data in and out of reconfigurable hardware.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

However, neither Fallside nor Smith disclose of a plurality of program data memories, each holding a program that creates a logic circuit directly in said reconfigurable hardware for each of said processing units, and an effective block selection unit that connects one of said program data memories to said processing element..

On the other hand, Hanrahan discloses of plurality of program data memories, each holding a program that creates a logic circuit directly in said reconfigurable hardware for each of said processing units (Figure 2A, entries akin to 32A, or alternatively, Figure 2B, each memory), and an effective block selection unit that connects one of said program data memories to said processing element (Figure 2A, Mux 34, or alternatively, the tri-state buffers of Figure 2B).

Hanrahan's teaching improves the efficiency of reconfigurable computer systems (col. 1, lines 43-44, explained in col. 1, lines 26-31).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Hanrahan would be able to be implemented into the invention of Fallside

and Smith, as Smith discloses in col.9, lines 5-11 of the concept of function prefetching in order to minimize the time cost associated with programmable logic resource configuration overhead, and Smith discloses of banks of memory in col. 4, lines 47-49 and chip selects in col. 6, line 17.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems.

36. **Consider claim 16**, Fallside discloses of an electronic computer including a processing device with reconfigurable hardware that can create a logic circuit for a processing unit (Figure 1, FPGA 104) and a control device (Figure 1, configuration control circuit 106) issuing an instruction to execute a command (col. 4, lines 27-29, initiate reconfiguration) when a processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions), said processing device including reconfigurable hardware (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA), a plurality of program data memories that hold programs for each said processing unit (Figure 3, PROM 204, RAM 208), creating logic circuits of the reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence), executing, by a control device the command execution instruction from the processing device (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence), switching the content

of a logic circuit executed by the reconfigurable hardware (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA), and an activate command controlling the effective block selection unit so as to make a specified program data memory effective and connecting it to the reconfigurable hardware (col. 6, lines 46-47, fpga_cs).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, and a program, wherein said program is generated, given a control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs, by executing a control flow analysis procedure for generating a command sequence executed by a control device, executing a command sequence implementation procedure for translating said command sequence into a data string, wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid, and executing a program data generation procedure for generating program data. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with. Furthermore, Fallside does not explicitly disclose of an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective.

On the other hand, Smith does disclose of switching and executing programs generated by dividing an application program into a plurality of processing units (col. 2,

lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data; col. 3, lines 1-6 for example disclose swapping of programmable logic configuration data between programmable logic resources and a secondary storage device), and a program (col. 12, line 1, final executable code image), wherein said program is generated, given a control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of the electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are

essentially input into the compiler and linker), by executing a control flow analysis procedure for generating a command sequence executed by said control device (col. 11, lines 57-63, compiling software functions into threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler; the control device is col. 5, lines 49-57, either arrangement that implements the VC-OS, which may be a microprocessor or a programmable logic device), executing a command sequence implementation procedure for translating said command sequence into a data string (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), and executing a program data generation procedure for generating program data (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility. It would have been readily recognized to

one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into configuration data, and Fallside discloses of an overall hardware environment in which a control device reconfigures a configurable logic device.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

However, neither Fallside nor Smith disclose of an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective.

On the other hand, Hanrahan discloses of an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective (Figure 2A, Mux 34, or alternatively, the tri-state buffers of Figure 2B, or alternatively the mux of Figure 8B), and an activate command controlling the effective block selection unit so as to make a specified program data memory effective and connecting it to the reconfigurable hardware (col. 4, lines 47-49, the control bit field determines what input is selected by multiplexer 30, or alternatively a part of an address as per Figures 2A and 2B).

Hanrahan's teaching improves the efficiency of reconfigurable computer systems (col. 1, lines 43-44, explained in col. 1, lines 26-31).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Hanrahan would be able to be implemented into the invention of Fallside and Smith, as Smith discloses in col.9, lines 5-11 of the concept of function prefetching in order to minimize the time cost associated with programmable logic resource configuration overhead, and Smith discloses of banks of memory in col. 4, lines 47-49 and chip selects in col. 6, line 17.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems.

37. **Consider claim 21**, Fallside discloses of an electronic computer including a processing device with reconfigurable hardware that can create a logic circuit for a processing unit (Figure 1, FPGA 104) and a control device (Figure 1, configuration control circuit 106) issuing an instruction to execute a command (col. 4, lines 27-29, initiate reconfiguration) when the processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions), a plurality of program data memories that hold programs for each said processing unit (Figure 3, PROM 204, RAM 208), creating logic circuits of the reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration

can commence), executing, by a control device the command execution instruction from the processing device (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence), switching the content of a logic circuit executed by the reconfigurable hardware (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA), and an activate command controlling the effective block selection unit so as to make a specified program data memory effective and connecting it to the reconfigurable hardware (col. 6, lines 46-47, fpga_cs).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, and a program, wherein said program is generated, given a control flow of the application program, completion data, structural information of an electronic computer and a plurality of command sets of the electronic computer as inputs, by executing a control flow analysis procedure for generating a command sequence executed by a control device, executing a command sequence implementation procedure for translating said command sequence into a data string, wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid, and executing a program data generation procedure for generating program data executed by the processing device. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with. Furthermore, Fallside does not explicitly disclose of an effective block

selection unit that selects one program data memory from the plurality of program data memories and that makes it effective.

On the other hand, Smith does disclose of switching and executing programs generated by dividing an application program into a plurality of processing units (col. 2, lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data; col. 3, lines 1-6 for example disclose swapping of programmable logic configuration data between programmable logic resources and a secondary storage device), and a program (col. 12, line 1, final executable code image), wherein said program is generated, given a control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of an electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of

programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are essentially input into the compiler and linker), by executing a control flow analysis procedure for generating a command sequence executed after a process (col. 11, lines 57-63, compiling software functions into threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler), executing a command sequence implementation procedure for translating said command sequence into a data string (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), and executing a program data generation procedure for generating program data executed by the processing device (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into configuration data, and Fallside discloses of an overall hardware environment in which a control device reconfigures a configurable logic device.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

However, neither Fallside nor Smith disclose of an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective.

On the other hand, Hanrahan discloses of an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective (Figure 2A, Mux 34, or alternatively, the tri-state buffers of Figure 2B, or alternatively the mux of Figure 8B), and an activate command controlling the effective block selection unit so as to make a specified program data memory effective and connecting it to the reconfigurable hardware (col. 4, lines 47-49, the control bit field

determines what input is selected by multiplexer 30, or alternatively a part of an address as per Figures 2A and 2B).

Hanrahan's teaching improves the efficiency of reconfigurable computer systems (col. 1, lines 43-44, explained in col. 1, lines 26-31).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Hanrahan would be able to be implemented into the invention of Fallside and Smith, as Smith discloses in col.9, lines 5-11 of the concept of function prefetching in order to minimize the time cost associated with programmable logic resource configuration overhead, and Smith discloses of banks of memory in col. 4, lines 47-49 and chip selects in col. 6, line 17.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems.

38. **Consider claim 5**, Fallside as modified by Hanrahan discloses that said control device interprets and executes (col. 6, lines 18-19, signals configuring FGPA); an activate command selecting one of said program data memories and activating said processing element (col. 6, lines 46-47, fpga_cs; also see Hanrahan); a halt command halting operation of said processing device (col. 6, lines 39-41; the command holds off

the configuration operation); a load_prg command transferring program data from a specified memory device to one of said program data memories (col. 6, lines 42-43, fpga_write); a cancel_prg command canceling a load_prg instruction (col. 6, lines 33-35, fpga_prog), and a wait_prg command waiting until completion of the load_prg instruction (col. 6, lines 44-45; the busy signal being asserted is analogous to the wait command).

39. **Consider claim 15**, Fallside and Smith do not disclose, after said switching, while a program in a predetermined program data memory is being executed, a next program is read into another program data memory.

On the other hand, Hanrahan does disclose, after said switching, while a program in a predetermined program data memory is being executed, a next program is read into another program data memory (see, for example, col. 1, lines 61-64, the arrangement into the background plane and the foreground plane allows a background plane to be loaded onto the chip without affecting the operation of the configuration state memory).

Hanrahan's teaching improves the efficiency of reconfigurable computer systems (col. 1, lines 43-44, explained in col. 1, lines 26-31).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that

the teaching of Hanrahan would be able to be implemented into the invention of Fallside and Smith, as Smith discloses in col.9, lines 5-11 of the concept of function prefetching in order to minimize the time cost associated with programmable logic resource configuration overhead, and Smith discloses of banks of memory in col. 4, lines 47-49 and chip selects in col. 6, line 17.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Hanrahan with the invention of Fallside and Smith in order to improve the efficiency of reconfigurable computer systems.

40. **Consider claim 17**, Fallside discloses said control device executes (col. 6, lines 18-19); a halt command halting the operation of said specified processing device (col. 6, lines 39-41; the command holds off the configuration operation); an interrupt command issuing an interrupt vector from said control device to said specified processing device (col. 6, lines 36-38; the signal triggers the start-up sequence which is analgous to the interrupt vector in the instant application; alternatively, lines 39-41 for an interrupt in general); a load_prg command transferring program data from a specified memory device to said program data memory (col. 6, lines 42-43); a cancel_prg command canceling a load_prg instruction (col. 6, lines 33-35), and a wait_prg command waiting until a completion of the load_prg instruction (col. 6, lines 44-45; the busy signal being asserted is analogous to the wait command).

41. **Consider claim 22**, Fallside discloses a procedure in which a halt command halting the operation of said specified processing device (col. 6, lines 39-41; the command holds off the configuration operation); an interrupt command issuing an interrupt vector from said control device to said specified processing device (col. 6, lines 36-38; the signal triggers the start-up sequence which is analgous to the interrupt vector in the instant application; alternatively, lines 39-41 for an interrupt in general); a load_prg command transferring program data from a specified memory device to said program data memory (col. 6, lines 42-43); a cancel_prg command canceling a load_prg instruction (col. 6, lines 33-35), and a wait_prg command waiting until the completion of the load_prg instruction (col. 6, lines 44-45; the busy signal being asserted is analogous to the wait command).

42. Claims 12, 14, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside et al. (Fallside) (US PAT 6326806) in view of Smith.

43. **Consider claim 12**, Fallside discloses a processing device (Figure 1 as a whole, specifically including the FPGA 104 and the configuration control circuit 106) including reconfigurable hardware that can create a logic circuit for each said processing unit (Figure 1, FPGA 104), and a control device (Figure 1, configuration control circuit 106) executing a command specified by the processing device (col. 4, lines 27-29, initiate reconfiguration); wherein said command is instructed to be executed when the processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions) and includes a command for

execution of switching programs logically creating the reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence); and said processing device comprises a second processing device including reconfigurable hardware that can create a logic circuit with a program (Figure 5, FPGA 1-2) and a second control device executing a command specified by the second processing device (Figure 1, configuration control circuit 106, or FPGA 1-2 itself).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of a device for dividing an application program into a plurality of processing units, and a program, wherein said program is generated, given a control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs, by executing a control flow analysis procedure for generating a command sequence executed after each process, executing a command sequence implementation procedure for translating said command sequence into a data string, wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid, and executing a program data generation procedure for generating program data. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with.

On the other hand, Smith does disclose of a device for dividing an application program into a plurality of processing units, (col. 2, lines 1-8, when an application is

compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data), and a program (col. 12, line 1, final executable code image), wherein said program is generated, given a control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of the electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are essentially input into the compiler and linker), by executing a control flow analysis procedure for generating a command sequence executed after each process (col. 11, lines 57-63, compiling software functions into

threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler), executing a command sequence implementation procedure for translating said command sequence into a data string (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), and executing a program data generation procedure for generating program data (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into configuration data, and Fallside discloses of an overall

hardware environment in which a control device reconfigures a configurable logic device.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

44. **Consider claim 14**, Fallside discloses an electronic computer including a processing device with reconfigurable hardware that can create a logic circuit for a processing unit (Figure 1, FPGA 104) and a control device (Figure 1, configuration control circuit 106), issuing an instruction to execute a command (col. 4, lines 27-29, initiate reconfiguration) when a processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions), and having a program (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA); and executing switching programs that logically create reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence) by a control device that has received the command execution instruction from the processing device (Figure 1, configuration control circuit 106).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, and a program,

wherein said program is generated, given a control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs, by executing a control flow analysis procedure for generating a command sequence executed after each process, executing a command sequence implementation procedure for translating said command sequence into a data string, and executing a program data generation procedure for generating program data. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with.

On the other hand, Smith does disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, (col. 2, lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data), and a program (col. 12, line 1, final executable code image), wherein said program is generated, given a control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of the electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe

Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are essentially input into the compiler and linker), by executing a control flow analysis procedure for generating a command sequence executed by said control device (col. 11, lines 57-63, compiling software functions into threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler; the control device is col. 5, lines 49-57, either arrangement that implements the VC-OS, which may be a microprocessor or a programmable logic device), executing a command sequence implementation procedure for translating said command sequence into a data string (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), and executing a program data generation

procedure for generating program data (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into configuration data, and Fallside discloses of an overall hardware environment in which a control device reconfigures a configurable logic device.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

45. **Consider claim 20**, Fallside discloses an electronic computer including a processing device with reconfigurable hardware that can create a logic circuit for a processing unit (Figure 1, FPGA 104) and a control device (Figure 1, configuration

control circuit 106), issuing an instruction to execute a command (col. 4, lines 27-29, initiate reconfiguration) when a processing device detects a predetermined condition (col. 4, lines 27-29; initiate reconfiguration in response to predetermined conditions), and having a program (Figure 1, FPGA 104; col. 2, lines 16-18, load a second configuration bitstream from the storage element into the FPGA); and executing switching said program that logically creates reconfigurable hardware (col. 7, lines 9-10, the CFG_TRIGGER signal indicates to the PLD that reconfiguration can commence) by said control device that has received the command execution instruction from the processing device (Figure 1, configuration control circuit 106).

However, although Fallside discloses of multiple logic circuits (Fallside, Figure 5), Fallside does not explicitly disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, and a program, wherein said program is generated, given a control flow of the application program, completion data, structural information of the electronic computer and a plurality of command sets of the electronic computer as inputs, by executing a control flow analysis procedure for generating a command sequence executed after each process, executing a command sequence implementation procedure for translating said command sequence into a data string, and executing a program data generation procedure for generating program data executed by the processing device. This is because Fallside's invention is directed toward *how* the FPGAs are configured but not *what* they are configured with.

On the other hand, Smith does disclose of switching and executing programs generated by dividing an application program into a plurality of processing units, (col. 2, lines 1-8, when an application is compiled, the functions of the application that are implemented in hardware are partitioned into blocks containing configuration data), and a program (col. 12, line 1, final executable code image), wherein said program is generated, given a control flow of the application program (col. 2, lines 18-20, col. 10, lines 51-53; software development tools; col. 11, lines 1-3; system design language profiler; note that col. 11, lines 1-3 discloses of analyzing critical paths; control flows are necessary in order to determine critical paths), completion data (col. 10, line 52, a set of constraints, further described in col. 11, lines 4-18; constraints may be thought of as completion data as constraints are data that dictate how the partitioning is ultimately completed), structural information of the electronic computer (col. 11, lines 11-14, resource library contains details about each available hardware resource) and a plurality of command sets of the electronic computer as inputs (col. 10, lines 49-53 describe Figure 6 as the process of compiling a high-level design specification or algorithm and executing it on a reconfigurable hardware architecture. In other words, the high-level design specification or algorithm is converted into either typical general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data as in col. 7, lines 1-4. For this conversion to occur, it is inherent that those general-purpose machine instructions or instructions which perform the run-time swapping of programmable logic device configuration data must be known of by the compilers and linker of Figure 7. Therefore, they are

essentially input into the compiler and linker), by executing a control flow analysis procedure for generating a command sequence executed by said control device (col. 11, lines 57-63, compiling software functions into threads using a HLL compiler and hardware functions into configuration patterns using a HDL compiler; the control device is col. 5, lines 49-57, either arrangement that implements the VC-OS, which may be a microprocessor or a programmable logic device), executing a command sequence implementation procedure for translating said command sequence into a data string (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler), wherein the command sequence procedure sets a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid (col. 11, lines 4-9, the partitioning phase takes into account timing relationships within functions and concurrent or sequence constraints between functions), and executing a program data generation procedure for generating program data executed by the processing device (col. 12, lines 1-6, linker and col. 11, lines 57-63, compiling hardware functions into configuration patterns using a HDL compiler).

The teaching of Smith allows for optimized execution times and parallelism for a computer handling a given application, as well as increased flexibility (Smith, col. 1, lines 44-48 and lines 56-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given

application, as well as increased flexibility. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Smith would be able to be applied to the environment of Fallside as Smith describes partitioning a program into configuration data, and Fallside discloses of an overall hardware environment in which a control device reconfigures a configurable logic device.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Smith with the invention of Fallside in order to enable optimized execution times and parallelism for a computer handling a given application as well as increased flexibility.

46. Claims 6-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside, Smith, and Hanrahan as applied to claim 1 above, and further in view of Birns et al. (Birns) (US PAT 5887189).

47. **Consider claim 6**, although Fallside discloses reading commands, interpreting, and executing it (col. 9, lines 12-20; the FPGA provides the desired configuration instruction *signals* to configuration control circuit and then triggers the reconfiguration, with the signals being CFG_MODE in col. 7, lines 3-5; also note that he also discloses that the configuration control circuit could be implemented as a microcontroller, col. 4, lines 56-57); Fallside nevertheless does not disclose a command code memory holding commands that said control device executes, wherein said control device comprises a command code reference device reading commands from the command code memory

according to an address specified by said processing device, interpreting, and executing it.

On the other hand, Birns does disclose a command code memory holding commands (Fig. 1, instruction memory 18) that said control device executes (col. 9, lines 49-51) wherein said control device comprises a command code reference device reading commands from the command code memory (col. 3, lines 27-31; decode unit) according to an address specified by said processing device (col. 9, lines 55-57), interpreting, and executing it (col. 9, lines 49-51).

It would have been readily recognized to one of ordinary skill in the art at the time of the invention that a control device such as a microsequencer that has instructions stored in memory that can be initiated when given an address is more configurable and cost effective than a control device which executes commands based on predefined signals and not addresses, as to compensate, each of the external devices issuing said signals would need to have additional hardware to implement a series of instructions.

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Birns with the invention of Fallside, Smith, and Hanrahan in order to increase configurability. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the teaching of Birns would be able to be applied to the invention of Fallside, as Fallside discloses the potential use of a microsequencer as a control device as noted above (col. 4, lines 56-57).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Birns with the invention of Fallside, Smith, and Hanrahan in order to allow greater configurability.

48. **Consider claim 7**, the claim is rejected for same reasons as claim 6 above. Furthermore, Fallside and Birns discloses that said command code reference device comprises an address counter holding the address of said command code memory (Birns, col. 9, line 56, program counter), and in the exchange of commands between said processing device and said control device (Fallside, col. 6, lines 18-19), a first address control line indicating that an address signal line outputted by said processing device is effective (Fallside, col. 6, line 17, output-enable signals; col. 7, lines 5-7; with the Mode Enable analogous to the Address enable), and a second address counter control line instructing whether the value of the address signal line is stored in the address counter as it is (Birns, col. 9, lines 66-67 and col. 10, line 1; absolute addresses) or the result of adding the value of the address signal line to the value of the address counter is stored in the address counter when the first control line is effective (Birns, col. 9, lines 55-57; relative branches and displacement).

49. Claims 8-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside, Smith, Hanrahan, and Birns as applied to claim 7 above, and further in view of Stewart et al. (Stewart) (US PAT 5473763).

50. **Consider claim 8**, Birns discloses said commands are stored in said command code memory in a format comprising a command code that classifies the commands (col. 3, lines 29-32; because the instructions are decoded, it is inherent that they are represented as some form of opcode), an address counter control code (col. 9, lines 66-67 and 55-57), and said address counter control code includes a load adr command setting the value of the address counter (col. 9, lines 66-67) and a add_adr command adding a specified value to the address counter (col. 9, lines 55-57).

However, Fallside, Smith, Hanrahan, and Birns do not explicitly disclose a flag that indicates whether or not the following command is executed.

On the other hand, Stewart does disclose a flag that indicates whether or not the following command is executed.

The use of a flag that indicates whether a following command is executed is a common way of putting a processor or microcontroller into idle mode (col. 7, lines 22-25) that doesn't require the use of repeated nop instructions, which typically lowers power consumption.

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Stewart with the invention of Fallside, Smith, Hanrahan, and Birns in order to save power. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the disclosed stop bit of Stewart fits into the environment of Fallside, Smith, Hanrahan, and Birns as the invention of Stewart deals with running certain program sequences at a starting address (col. 3, lines 26-32) upon the activation of an external interrupt trigger (col. 3, lines 38-

40), which is analogous to Fallside, Smith, Hanrahan, and Birns running certain program sequences upon the receipt of an address and enabling signal from an external reconfigurable logic.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Stewart with the invention of Fallside, Smith, Hanrahan, and Birns in order to save power.

51. **Consider claim 9**, Birns discloses said address counter control code includes a push_adr command that hides the address counter in an address counter stack provided in said control device and that sets a new value to the address counter, and a pop_adr command that returns the value of the address counter stack to the address counter (both of these commands are inherent in col. 10, lines 2-3, return address stack).

52. Claims 10 and 11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside, Smith, and Hanrahan as applied to claim 1 above, and further in view of Sachs et al. (Sachs) (US PAT 4860192).

53. **Consider claim 10**, Fallside, Smith, and Hanrahan do not disclose a cache device including a cache memory that temporarily holds data to be transferred to said processing device and a cache controller that controls the cache memory wherein the cache controller is controlled by a command issued by said processing device.

On the other hand, Sachs does disclose a cache device including a cache memory (col. 1, line 18, cache memory) that temporarily holds data to be transferred to said processing device (col. 1, lines 37-41, cache memory) and a cache controller that controls the cache memory wherein the cache controller is controlled by a command issued by said processing device (col. 1, line 18, cache controller).

It would have been readily recognized to one of ordinary skill in the art at the time of the invention that implementing a cache in general allows for faster memory accesses, leading to accelerated data transfer and reduced execution time.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Sachs with the invention of Fallside, Smith, and Hanrahan in order to reduce total execution and configuration time.

54. **Consider claim 11**, the claim is rejected for same reasons as claim 10 above. Furthermore, Sachs discloses said cache device comprises an address translation device that translates an address defined externally to said processing device into an address defined inside of the processing device, and the address translation device is controlled by a command issued by said processing device (col. 1, lines 19, 32-40; the externally defined address is the main memory, the internally defined address is the cache).

55. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Fallside and Smith as applied to claim 12 above, and further in view of Abramovici (US 6034538).

56. **Consider claim 13**, Fallside and Smith do not disclose a semiconductor integrated circuit implementing the electronic computer as defined in claim 1.

On the other hand, Abramovici does disclose a semiconductor integrated circuit implementing the electronic computer as defined in claim 1 (col. 4, lines 39-40)

It would have been readily recognized to one of ordinary skill in the art at the time of the invention that Implementing an electronic computer on a semiconductor integrated circuit is an optimal method of doing so for both space and performance considerations.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Abramovici with the invention of Fallside and Smith because of space and performance considerations.

Response to Arguments

57. Applicant argues on pages 12-15 that the prior art does not teach the newly added limitation of setting a time period for loading configuration data of the reconfigurable hardware or making the configuration data valid. However, as cited in the rejection above, Smith does teach the aforementioned limitation.

Conclusion

58. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

h. Trimberger (US 6105105) discloses of sequencing logic connecting to configuration select circuits which can take in parameters from the programmable logic.

i. Master et al. (US 6836839 B2) discloses of timing and scheduling configuration and reconfigurations of heterogeneous computational elements.

j. Uriu et al. (US 20060004993) discloses of calculating execution times of configuration by a reconfigurable processor.

k. Casselman (US 5684980) discloses of FPGA reconfiguration in response to instructions.

l. Vorbach et al. (US 6571381) discloses of automatic configuration and reconfiguration of modules through reconfiguration requests.

59. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any

extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

60. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Keith Vicary whose telephone number is (571)270-1314. The examiner can normally be reached on Monday - Thursday, 6:15 a.m. - 5:45 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 571-272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Richard Ellis/
Primary Examiner, Art Unit 2183

/Keith Vicary/
Examiner, Art Unit 2183